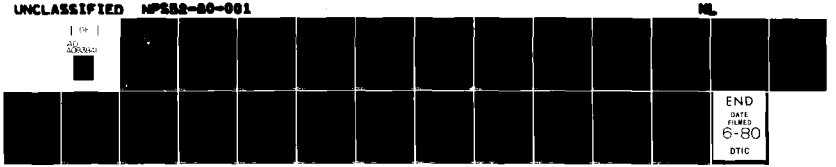
AD-A083 841 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
THE TEXT EDITOR AS A UNIFORM MAN/MACHINE INTERFACE. A PROPOSAL --ETC(U)
FEB 80 L A COX
UNCLASSIFIED NPS52-80-001 NL



END
DATE FILMED
6-80
DTIC

NPS52-80-001

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THE TEXT EDITOR AS A UNIFORM MAN/MACHINE INTERFACE. A PROPOSAL FOR A STANDARD EDITOR

Lyle Ashton Cox, Jr.

February 1980

26 Feb 1980

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral J. J. Ekelund
Superintendent

Jack R. Borsting
Provost

The work reported herein was supported in part by the Foundation Research Program of the Naval Postgraduate School with funds provided by the Chief of Naval Research.

This report was prepared by:

LYLE A. COX, JR.
Assistant Professor of Computer Science

Reviewed by:

G. H. BRADLEY, Chairman
Department of Computer Science

Released by:

WILLIAM M. TOLLES
Dean of Research

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: NPS52-80-001	2. GOVT ACCESSION NO.: AD-A083 841
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): The Text Editor as a Uniform Man/Machine Interface, A Proposal for a Standard Editor	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): Lyle Ashton Cox, Jr.	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, CA 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61152N; RR000-01-10; N0001480WR00054
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, CA 93940	12. REPORT DATE: February 1980
	13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Text Editor
Software Engineering
Computer Networks

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

There is a substantial group of professionals, scientists, engineers, and managers who are justifiably reluctant to use computer networks such as the ARPANET. This phenomenon continues despite the fact that they recognize some of the benefits of computational assistance, that they have had experience using computer systems, and that they have access to such a network. Their reluctance usually stems from the feeling that the very machines and systems is not justified by the occasional or intermittent nature of their computational

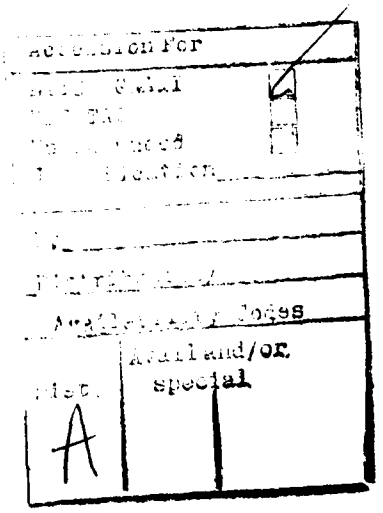
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 | 1

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

problems. In learning to use a new system, a large part of the familiarization effort is spent in trying to learn to use a new text editing program. If such a utility program were standardized and made available on all of the machines on the network, a large obstacle to the efficient use of such systems might be removed. The design of such a system, a Standard Line EDitor called "SLED", is proposed here.

ABSTRACT

There is a substantial group of professionals, Scientists, engineers, and managers who are justifiably reluctant to use computer networks such as the ARPANET. This phenomenon continues despite the fact that they have had experience using computer systems, and that they have access to such a network. Their reluctance usually stems from the feeling that the very great effort required to familiarize themselves with new machines and systems is not justified by the occasional or intermittent nature of their computational problems. In learning to use a new system, a large part of the familiarization effort is spent in trying to learn to use a new text editing program. If such a utility program were standardized and made available on all of the machines on the network, a large obstacle to the efficient use of such systems might be removed. The design of such a system, a Standard Line EDitor called "SLED" is proposed here.



The Text Editor as A Uniform Man/Machine Interface
A Proposal for a Standard Editor

by

Lyle Ashton Cox Jr.

Department of Computer Science
Naval Postgraduate School
Monterey, California

ABSTRACT

There is a substantial group of professionals, scientists, engineers, and managers who are justifiably reluctant to use computer networks such as the ARPANET. This phenomenon continues despite the fact that they recognize some of the benefits of computational assistance, that they have had experience using computer systems, and that they have access to such a network. Their reluctance usually stems from the feeling that the very great effort required to familiarize themselves with new machines and systems is not justified by the occasional or intermittent nature of their computational problems. In learning to use a new system, a large part of the familiarization effort is spent in trying to learn to use a new text editing program. If such a utility program were standardized and made available on all of the machines on the network, a large obstacle to the efficient use of such systems might be removed. The design of such a system, a Standard Line EDitor called "SLED" is proposed here.

BACKGROUND

The United States Navy is currently conducting an experiment to determine the effectiveness of computer networking in providing the computing support required by the Navy laboratories. In the course of this experiment, significant resources of the laboratories are being organized into a "Navy Laboratory COmputer Network" or "NALCON" (1) (2) to promote the efficient use of the physical and logical resources of the Navy laboratories.

While the NALCON system was being implemented, it was recognized that software technology often poses greater problems than does the hardware design and construction. In response to this, the Navy Laboratory Computer Committee (NLCC) formed a Software Technology Working Group. It is the goal of this group to address the specific software problems of NALCON, and to consider the larger problems of software technology in the Navy computing community.

After a series of meetings, the Software Technology Working Group reported (3) that the number and diverse nature of text editor programs constituted a significant obstacle (both real and psychological) to the efficient use of network resources. It was suggested that either a standard editor be developed, or that all network computers contain editors with a standard subset of commands.

Subsequently, work was undertaken at the Naval Postgraduate School to determine the desirable characteristics which such an editor should display, and to define or specify the user interface for this proposed standard editor. The results of this effort are described below.

PHILOSOPHY

Before we describe the proposed standard editor, it is appropriate that several non-technical, philosophical, aspects of the editor design be discussed.

It should be recognized that any editor will certain some characteristics which significant portions of the computer user community will find objectionable. The implementation of text editor features is often a matter of personal taste. You can not please all of the people all of the time. We believe that the proposed editor is well thought out, and is based upon the experience of thousands of computer users spanning well over a decade of network and timesharing experience. Our solution is certainly not unique. There are other solutions. We have confidence that ours is one of the better possibilities, and should seriously be considered as a standard.

With regards to the specification itself, in the following sections we will attempt to describe the editor informally, using a mixture of two techniques. The description will not be a rigorous specification in the software engineering sense; nor will it be a "users view" of the editor. By combining aspects of

these two techniques - specifications and a "users introduction" type description of the editor - we hope to both describe the editor in sufficient detail that it can be implemented, and give the readers a "feel" for using the editor. We ask that the reader keep in mind the goal of the standard editor project, and this philosophy while reading the remainder of this paper.

THE GOALS OF A "STANDARD EDITOR"

The goal of the Standard Editor project is to define and develop a simple and easy to use text editor program which can be readily implemented on a wide variety of host machines and operating systems. This objective and the computer network context of its development allow further definition of SLED requirements.

The usage envisioned for SLED is twofold: casual use by persons local to the host; and use by occasional network guests of the host. Such a simple, basic editor can not and should not attempt to replace the principal system editor programs available on the host machines. Since the scope of usage is thus reduced, no attempt need be made to design SLED to be all powerful (and hence complex). SLED only needs to support the basic text editing functions, and if these functions are supported in a well designed, well documented, easy to use implementation, the basic goals of the project can be fulfilled.

The limited scope also allows implementations to be

accomplished without undue attention to questions of execution efficiency which are vital in the design of a principal system editor. Thus implementations can be realized using higher level computer programming languages, with emphasis on portability and system independence.

The wide variety of users anticipated, and the large number of implementations, requires several characteristics of the editor to be present in all its implementations. Certainly all the implementations must be as nearly uniform (in terms of the user interface) as current software technology permits. In designing the editor, the usage and commands must be kept simple. The commands must function "intuitively" and be easy to learn and remember. Further, no special terminal character set or line speed assumptions can be made.

SPECIFYING THE EDITOR

From these general requirements, a more detailed specification was developed. Great effort was made to keep the editor simple (from the user's point of view). Of secondary importance was machine and system independence, and portability of the implementations. The resultant specifications are described informally (from the point of view of the user) below.

The constraints upon terminals and line speeds, coupled with the restricted nature of the editor led us to select a "line oriented editor" approach. With the addition of a logical line terminator character and a simple display function, line

oriented editors have been shown to function efficiently with a variety of terminals using a wide range of line speeds in a time sharing environment (4),(5).

A minimum set of commands consistent with easy use has been selected (5),(6). These commands draw their mnemonic symbols from the first letters of their key words taken in normal English language order. For example, the command to "insert text <A>fter <L>ine 5" is "AI5". There was some desire to limit the mnemonics to single characters. While this would decrease the total number of key strokes required in an edit session, the decrease is a small percentage of the total number of characters typed. The advantages of natural English language command ordering combined with the relative independence from many other editors which use single character commands (hence decreasing the chances of confusion and error for persons inadvertently reverting to other editor commands) was considered to outweigh the advantages of exclusive use of single character mnemonics.

There are a total of eleven commands, only seven of which need be used to obtain full text editing capability. These eleven commands can be roughly subdivided into five basic groups:

1. Line Insertion and Replacement commands (two commands)
2. String Replacement commands (one basic command)
3. String Search commands (one basic command)
4. Terminal Output commands (four commands)
5. Control commands (three commands)

These commands are more fully described in Table 1.

Since many of the prospective SLED users are only occasional users of computer systems, initiation of all implementations should be uniformly commenced by typing only "SLED<Carrage-return">" after linking and logging into the network host machine. At this point, the casual user may ask for a menu to refresh his memory as to the basic commands and their format via the "M" command. (See Figure 1 for an example of the <M>enu output.) The possibility of SLED users equipped with low speed terminals or low speed lines requires that this message be kept brief. Two incorrectly formatted requests in series will automatically cause the execution of a <M>enu command.

A more experienced user may directly execute the <V>ersion command which will print a brief version identification, the name and telephone numbers of consultants who can provide some help if required, and will explicitly identify any features of the editor which are required by the local system. A sample of the <V>ersion output is shown in Figure 2.

The editor, like many other editors, features essentially two modes: "Edit command" mode and "text Insert" mode. SLED, when initialized, starts in "Edit command" mode, and requests an edit command from the user's console by transmitting to the user the prompt "E>". This prompt will also be transmitted following the successful execution of any edit command message line (except one containing the "<Q>uit" command) and the editor will await further instructions. Following this prompt the user can enter any command shown in Table 1. Commands and their fields

can be separated by any valid logical message terminator (see Figure 2). The character "Carriage return" always serves as a message terminator. One other character is provided for use as a message terminator, and this is changible at the direction of the user via the <C>hange <T>erminator command. This feature allows the "stacking" of editor commands within a single physical line of input. This feature is demonstrated below, and is extremely convenient in a network operating environment. If the command executed by the user causes the editor to enter the "text Insert" mode, the editor will prompt the user for data with the symbol "I>". All text entered after these prompts will be copied directly to the text file, and will not be interpreted as edit commands. The only way to return from the "text Insert" mode to the "Edit command" mode is by entering a single message consisting of ONLY a period ".".

While these two prompts are somewhat inconvenient to users desiring to operate in a "non-echo" mode, they are, in general, necessary for two reasons. They are useful in confirming to the (inexperienced) user which mode he is currently in; and they are vital in systems which do not save data buffers as a synchronization mechanism (for example a PDP-11 using the RSX-11 operating system which does NOT allow "type ahead" of logical read commands).

These modes and their functions can be better understood from the following example of SLED usage:

EXAMPLE 1: USING SLED TO CREATE A FILE
(From a UNIX Like implementation)

```
%SLED
E>O
filename?>NEW.file
-creating file "NEW.file"-
E>AL@
I> First text line
I>Second text line
I> Third text line
I>.
F>L1,3
  1  First text line
  2 Second text line
  3  Third text line
F>Q
%
```

In the above example, the SLED editor was used to create a new text file, and to enter three simple lines of text. Use of the logical message terminator key (if available) can significantly simplify the use of the editor. This key allows several command lines (either edit commands or insert lines) to be entered on a single physical line from the terminal. Example 2 shows the use of the logical message terminator key (shown as "$") to "stack" several editor commands into a single line of input. The effect of the commands shown in this example are the same as those shown in Example 1.

EXAMPLE 2: THE LOGICAL MESSAGE TERMINATOR
(Same effect as Example 1)

```
%SLED
E>O$NEW.file$AL@
-creating file "NEW.file"-
I> First text line$Second text line
I> Third text line$.$L1,3$Q
  1  First text line
  2 Second text line
  3  Third text line
%
```

Note that in the fifth line of the above example, a line of text was inserted, and then the insert mode was terminated (by sending a message consisting of only a "." delimited by the logical message terminator) and "Edit mode" commands were then sent. It is this power of transmitting multiple messages delimited by a reserved key which allows those users with slow terminals or those experiencing long data transmission delays over the network circuits to effectively use the editor.

Many persons considering the editor instruction set in Table 1 ask how it is possible to delete lines and patterns with this editor. These functions are acomplished with the use of the "replace" functions, replacing the items to be deleted with "nothing". In the context of this editor, a "string" consists of a sequence of characters which does not contain any of the logical message terminators. Thus, a string is a portion of a line since all lines terminate with a <RETURN> which is a terminator. To replace portions of lines one uses the <R>eplace <S>tring command, while the <R>eplace <L>ine command is used for larger modifications. For example, consider the file created in Example 2: supppose we wanted to delete the string "text" in the first line, and delete the entire third line. Using the "RS" and "RL" commands this could be accomplished as shown in Example 3.

EXAMPLE 3: DELETING STRINGS AND LINES

```
%SLED
E>O$NEW.file
- 3 lines in file "NEW.file"-
E>L1
  1  First text line
E>RS$text$$L1
  1  First  line
E>RL3$.$S1
  1  First  line
  2 Second text line
E>Q
%
```

In the above example, the pattern to be deleted in line number one was replaced with a string consisting of no characters. The third line was deleted by replacing it with a null line (ie. entering INSERT mode, and exiting -via a message consisting of only a period- without entering any data). In much the same manner the desire to insert text "Before Line n" can be accomplished with commands to insert text "After Line n-1".

SLED implementations must also cushion the user from his mistakes (ie. provide "fail soft" features). For example, an attempt to open a non existant file should produce an error message as shown in Example 4.

EXAMPLE 4: Soft Failure

```
%SLED
E>L1
-no text file open-
E>O$NEW.file
 2 lines in file "NEW.file"
E>Q
%
```

In addition to the normal demands of defensive programming,

the nature of the editor requires that careful attention be paid to error detection and recovery. The variety of terminal types and line speeds requires that the SLED messages be both clear and concise. The minimum set of SLED adviscry messages is shown in Table 2, along with the circumstances which will cause them to be generated. All implementations must detect these situations and recover gracefully. Individual implementations are encouraged to perform more sophisticated consistency checks for abnormal user and system conditions.

CONCLUSIONS

Significant increases in the productivity of computing professionals can be realized if we can make our existing computer resources more "usable". The ultimate goal -the creation of an effective and uniform man/computer interface- is perhaps idealistic and unachievable. There are however, a number of areas where we can achieve success, and make more efficient usage of our human and machine resources. One such area is uniformity of software development tools in distributed computing systems.

The Standard Line EDitor project is an exreriment in this vein. Several testbed implementations of SLED are now being written. With the completion of these programs, and with the continued support of the Navy Laboratory Computer Committee and the NLCC Software Technology Working Group, it is hoped that SLED may be implemented throughout the NALCON. This will be a

first, small, experimental step along the pathway to developing effective and uniform software development tools for use in computer network environments.

Table 1:

SLED Command Summary

COMMAND FUNCTION	USAGE
LINE INSERTION AND REPLACEMENT COMMANDS:	
insert text After Line number n	ALn (*)
insert text to Replace Lines n thru m	RLn,m (*)
or.. to insert text to Replace a single Line, n	RLn (*)
STRING REPLACEMENT COMMANDS	
Replace all occurrences of String "p" with the string "q" in lines n thru m inclusive.	RSn,mpq$
or.... Replace all occurrences of String "p" with string "q" in line n	RSnpq$
or.... Replace all occurences of String "p" with string "q" in the current line.	RSpq$
STRING SEARCH COMMANDS	
Display all lines containing String "p"	ISp
or... Display all lines between lines n and m inclusive which contain string "p"	DSn,mp
OUTPUT COMMANDS	
set current Line to n and display that line	In
or... display the current line	L
or... display Lines n thru m inclusive, and set the current line to m.	In,m
display a "Screen" of lines (20 lines) beginning with line n	Sn
or... display a "Screen" beginning with the current line	S
display Version and implementation information	V
display <M>enu of editor commands	M
CONTROL COMMANDS	
Open or Create a file for editing	O
Quit the editing session and update all files	Q
Change the message Terminator	CT

The symbol "$" indicates the use of a logical message terminator, not the <RETURN> key. (See further the text and Figure 2.)

(*) These commands cause the editor to enter the TEXT INSERT mode.

Figure 1:
Sample output from <M>enu command

```
                    SLED COMMAND SUMMARY
LINE/TEXT INSERT                        STRING REPLACEMENT
 ALn     insert <A>fter <L>ine n         RS$p$q$   <R>eplace <S>tring
 RLn     <R>eplace <L>ine n .or.         RSn$p$q$     "p" with "q" in
 RLn,m       lines n thru m              RSn,n$p$q$  inidcated lines.
OUTPUT COMMANDS                         STRING SEARCH
 L       display current<L>ine           DS$p$     <D>isplay lines wt
 Ln      or line n,                                  string "p",
 Ln,m    lines n thru m                  DSn,m$p$  or show any lines
 S       <S>how a "screen" of lines            n-m containing "p"
 Sn      show a screen about ln #n       CONTROL COMMANDS
 M       show command <M>enu (this)      O          <O>pen a file or
 V       show <V>ersion information         create a file for editing
                                         CT  <C>hange the logical
  TO <Q>UIT TEE EDIT TYPE "Q<ret>"       message <T>erminator
```

Figure 2:
Sample output from <V>ersion Command

```
          SLED Version Pas1.1 NPS-Monterey 800503
     Local Expert is J. Doe 408-646-2449 0800-1600 PST

     LINE DELETE KEY is <CONTROL-U>
     CHARACTER DELETE KEY is <RUBOUT>  (also called DELETE)

     EDITOR LOGICAL MESSAGE TERMINATORS ARE:
          (1) <RETURN>     and     (2) <ESCAPE> (echos "$")
** the message terminator can NOT be changed to "Line-Feed" **

     Local System supports UPPER/lower case
     Local System DID NOT require deviation fm SLED standard.
```

Table 2:

SLED Minimum Advisory Messages

Message	Situation and Editor response
-invalid command-	in command mode an unrecognizable command was issued (or an ambigious delimiter was used.)
-n lines in file "f"-	an <O>pen command on file "f" completed successfully.
-creating file "f"-	tried to <O>pen file "f", which did not exist. A new file is created.
-closing file "f"-	an <O>pen command was issued with a file already open. This file, "f" is updated and closed, and the command proceeds normally.
-no text file open-	an edit command was executed without a text file open.
-no string found-	a "RS" or "DS" command was issued, and the search string was not found.
old string?>	(A prompt) A "RS" or "DS" command was issued and the first string was not specified. The editor now waits for the string to be entered
new string?>	(A prompt) A "RS" command was issued where the second string was not specified. The editor now waits for the string to be entered.
file name?>	(A prompt) An <O>pen command was issued without specifying the name of the file. The editor now waits for the file name to be entered.
terminator?>	(A prompt) A <C>hange <T>erminator comman was issued. A valid ASCII character is entered to act as a message terminator.

REFERENCES

(1) "The Navy Laboratory Computer Network (NAICON)", I. Larry Avrunin, David Taylor Naval Ship R & D Center Report, 1979.

(2) "ARPANET Information Brochure", Defense Communications Agency August, 1976.

(3) NLCC Software Technology Working Group Memorandum of 16 November, 1979.

(4) "TRIX Report" A. Cecil and H. Moll, University of California LLL Report UCID-30100 Rev. 2, December, 1976.

(5) "TRIX AC Text Editor User's Manual" A. Cecil and H. Moll, University of California LLL Report UCID-30040 Rev. 9, July 1977.

(6) "ED (I)" K. Thompson and D. Richie in "UNIX Programmer's Manual" Bell Laboratories, May, 1975.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Office of Research Administration Code 012A Naval Postgraduate School Monterey, California 93940	1
4.	Chairman, Code 52Bz Computer Science Department Naval Postgraduate School Monterey, California 93940	30
5.	Lyle A. Cox, Jr., Code 52Cl Computer Science Department Naval Postgraduate School Monterey, California 93940	10
6.	I. Larry Avrunin David Taylor Naval Ship Research and Development Center (Code 18) Carderock Laboratory Bethesda, MD 20084	1
7.	R. P. Crabb, Code 9134 Naval Ocean Systems Center San Diego, CA 92152	3
8.	G. H. Gleissner David Taylor Naval Ship Research and Development Center (Code 18) Carderock Laboratory Bethesda, MD 20084	1
9.	Kathryn Heninger, Code 7503 Naval Research Lab Washington D.C. 20375	3
10.	Ronald P. Kasik, Code 4451 Naval Underwater Systems Center Newport, RI 02840	3

11.	Commander, Code 503 Naval Air Development Center Warminster, Pennsylvania 18974	3
12.	Mark Underwood, Code P204 NPRDC San Diego, CA 92152	3
13.	Michael Wallace, Code 1828 DTNSRDC Bethesda, MD 20084	3
14.	Walter P. Warner, Code K70 NSWC Dahlgren, VA 22448	3
15.	John Zenor, Code 31302 Naval Weapons Center China Lake, CA 96555	3